Universität Bremen

Datenbanksysteme WS2008/9

VAK 03-05-H-703.01

Hausarbeit („Datenbankentwurf“)

Tilman de Bruin <tilman@informatik.uni-bremen.de>
7. Semester Informatik (Diplom)
Matrikel-Nr.: 205 17 17

Die vorliegende Hausarbeit wurde im Rahmen der Lehrveranstaltung „Datenbanksysteme“ (bei Martin Gogolla) im WS2008/9 an der Universität Bremen angefertigt.

Die Lehrveranstaltung behandelte das Thema *Datenbanksysteme* von der historischen Entwicklung sowie den Aufgaben und der Architektur solcher Systeme, über unterschiedliche Datenmodelle (Entity-Relationship-Modell, Relationen-Modell, Netzwerk-Modell, Hierarchisches-Modell, objektorientierte Datenmodelle, semistrukturierte Datenmodelle) und Datenbank-Sprachen (RA, BK, TK, SQL, QUEL, QBE), bis zu Programmierschnittstellen, Datenintegrität und Datenschutz, und schließlich dem Standardverfahren des relationalen Datenbankentwurf. Der Übungsbetrieb wurde betreut von Mirco Kuhlmann.

Zweck dieser Hausarbeit ist es, am konkreten Beispiel, den Entwurfsprozeß einer Datenbank im Detail nachzuvollziehen.

Inhaltsverzeichnis

1 Einordnung

In diesem Kapitel werden zunächst grundlegende Begriffe der Datenbanktheorie kurz erklärt. Anschließend wird der Zweck der zu entwerfenden Datenbank, sowie ihre Einbettung in den entsprechenden „Weltausschnitt“ erläutert.

1.1 Begriffsklärung

Ein **Datenbanksystem (DBS)** besteht aus einer **Datenbank (DB)** und einem **Datenbank-Managementsystem (DBMS)**. Die DB realisiert die einheitlich strukturierte Speicherung aller verwalteten Daten, und das DBMS sorgt für eine zentrale Zugriffsschnittstelle auf diesen Datenbestand.

Grundsätzlich unterscheidet man zwischen den eigentlichen, in einem System verwalteten, **Daten** und den, die Struktur (Typen, Zusammenhänge) dieser Daten beschreibenden, **Metadaten**. Erstere spiegeln sich wider im *DB-Zustand*, letztere im *DB-Schema*.

Die Forderung nach *logischer*[1] und *physischer*[2] **Datenunabhängigkeit** führte zu der sog. **Drei-Ebenen-Architektur** (auch *ANSI-SPARC-Architektur*[3]).

In der Drei-Ebenen-Architektur werden **interne**, **konzeptionelle** und **externe Ebene** unterschieden. Gegenstand der internen Ebene ist, einen möglichst performanten Zugriff auf den möglichst effizient organisierten Datenbestand bereitzustellen; sie wird in dieser Ausarbeitung nicht weiter thematisiert. Die konzeptionelle Ebene realisiert eine logisch einheitliche Gesamtsicht auf den Datenbestand. Angestrebt werden Vollständigkeit (nur was in der konzeptionelle Ebene berücksichtigt wurde, ist später als Funktionsumfang verfügbar), Redundanzfreiheit (welche durch *Normalisierung* zu erreichen versucht wird) und Vermeidung von Anomalien. Die externe Ebene schließlich erlaubt, für verschiedene Verwendungen derselben DB unterschiedliche (eingeschränkte, oder kombinierte) *Sichten* auf den Datenbestand zu definieren, so daß den individuellen fachlichen Anforderungen einer Anwendung jeweils exakt entsprochen werden kann (was auch *Datenschutz* mit einschließt).

Nachfolgend werden die in der Vorlesung eingeführten Fachbegriffe als bekannt vorausgesetzt und ggf. bei Gebrauch nicht neu definiert.

[1] Unter logischer Datenunabhängigkeit versteht man eine Stabilität der Benutzungsschnittstelle gegen Änderungen an der Datenbankstruktur.

[2] Physische Datenunabhängigkeit bedeutet, daß eine Änderung zugrundeliegender Algorithmen und Datenstrukturen keine Änderungen an der Datenbankstruktur erfordert.

[3] Das Konzept wurde 1975 vom **S**tandards **P**lanning **a**nd **R**equirements **C**ommittee des **A**merican **N**ational **S**tandards **I**nstitute entwickelt.

1.2 Zweck

Zweck der zu entwerfenden Datenbank ist es, eine Art enzyklopädisches Lexikon bzw. eine Wissensbasis (ähnlich dem Nachschlagewerk WIKIPEDIA[4]) bereitzustellen. Es werden also bestimmte Stichwörter mit jeweils passenden Artikeln in Beziehung gesetzt.

Anders als WIKIPEDIA verfolgt das zu modellierende System aber einen allgemeineren Anspruch: Ausgehend von einer gemeinsamen, globalen Datenbasis sollen sich (mithilfe von „Filtern“) unterschiedlichste konkrete Anwendungen verwirklichen lassen (wie z. B. diverse Nachschlagewerke, Wörterbücher, Kochbücher, DIY-Sammlungen, u. v. m.).

1.3 Kontext

Die Idee ist, daß dem Benutzer zur Interaktion mit dem System eine Suchmaske angeboten wird, in der er das gesuchte Stichwort, sowie einen – dem gewünschten Thema entsprechenden – „Filter“ einstellen kann.

Das System liefert dann als Antwort die zur Suchanfrage passenden Artikel aus der zugrundeliegenden Datenbasis.

[4] www.wikipedia.org

2 Struktur der Daten

Dieses Kapitel stellt in knappem Umfang die Anforderungen an das System dar und entwickelt daraus, zu dessen Umsetzung notwendige, Konzepte und deren Beziehungen. Schließlich werden die zuvor gewonnenen Informationen in einem formalen Modell zusammengefaßt.

2.1 Übersicht

Damit das Ziel – eine globale Datenbasis, die sich in diverse Anwendungen spezialisieren läßt – erreicht werden kann, müssen einige Anforderungen erfüllt sein:

- Sowohl die *Stichwörter*, als auch die *Artikel* sind zu repräsentieren. Ggf. muß außerdem jeweils eine *Hierarchie* der repräsentierten Inhalte (im Sinne einer Taxonomie) unterstützt werden.
- Um *Metainformationen* (wie Quellennachweise, Relevanzbetrachtungen, Kritik, usw.) angeben zu können, ist es sinnvoll je Stichwort bzw. Artikel einen Diskussionsbaum anzulegen.
- Die Stichwort/Artikel-*Zuordnungen* sind zu repräsentieren.
- Um die o. g. „Filter“-Funktionalität umzusetzen ist es nötig, daß erstens die Stichwort/Artikel-Zuordnungen keine 1-zu-1, sonden *n-zu-m Beziehungen*, sind, und zweitens *Kontexte* repräsentiert werden, in denen die Zuordnungen jeweils gelten.
- Vermutlich ist es wünschenswert, daß die „Geltung“ einer Stichwort/Artikel-Zuordnung (je Kontext) von jedem Benutzer individuell unterschiedlich bewertet werden kann (da verschiedene Betrachter unterschiedliche Zusammenhänge als gültig bzw. passend beurteilen) – dazu müßte das System eine *Nutzerverwaltung* beinhalten. Von dieser Funktionalität wird aber in der vorliegenden Ausarbeitung abgesehen.

2.2 Konzepte und Beziehungen

Es folgt eine Übersicht der wesentlichen Konzepte und Beziehungen.

Stichwort: Ein Stichwort ist eine einfache Zeichenkette.

Artikel: Ein Artikel ist ebenfalls eine Zeichenkette (evt. einer bestimmten Auszeichnungssprache).

Metainformation: Metadaten bestehen sinnvollerweise aus einem Nutzernamen, einem Datum und dem eigentlichen Inhalt.

Kontext: Der Kontext kann im Grunde eine beliebeige Entität sein, da es nur darauf ankommt, daß er für alle Angaben zu einer Thematik konstant bleibt (um später eben den „gemeinsamen Kontext“ dieser Angaben identifizieren zu können).

hierarchie: Vom Stichwort-Typ besteht eine funktionale Beziehung zu sich selbst, um eine Taxonomie zu ermöglichen.

hierarchie: Vom Artikel-Typ besteht ebenfalls eine funktionale Beziehung zu sich selbst, aus analogen Gründen.

diskussionsbaum: Zwischen Metainformation-Typ und Stichwort-Typ besteht eine Beziehung, weil die Wurzel eines Diskussionsbaum einem konkreten Stichwort zugeordnet ist.

diskussionsbaum: Zwischen Metainformation-Typ und Artikel-Typ besteht eine Beziehung, wiederum analog zum Stichwort-Typ.

hierarchie: Vom Metainformation-Typ besteht eine funktionale Beziehung zu sich selbst, um den Diskussionsbaum realisieren zu können.

zuordnung: Stichwort-Typ, Artikel-Typ, sowie Kontext-Typ stehen miteinander in Beziehung, weil so die System-Kernfunktionalität ermöglicht werden kann.

2.3 Formales Modell

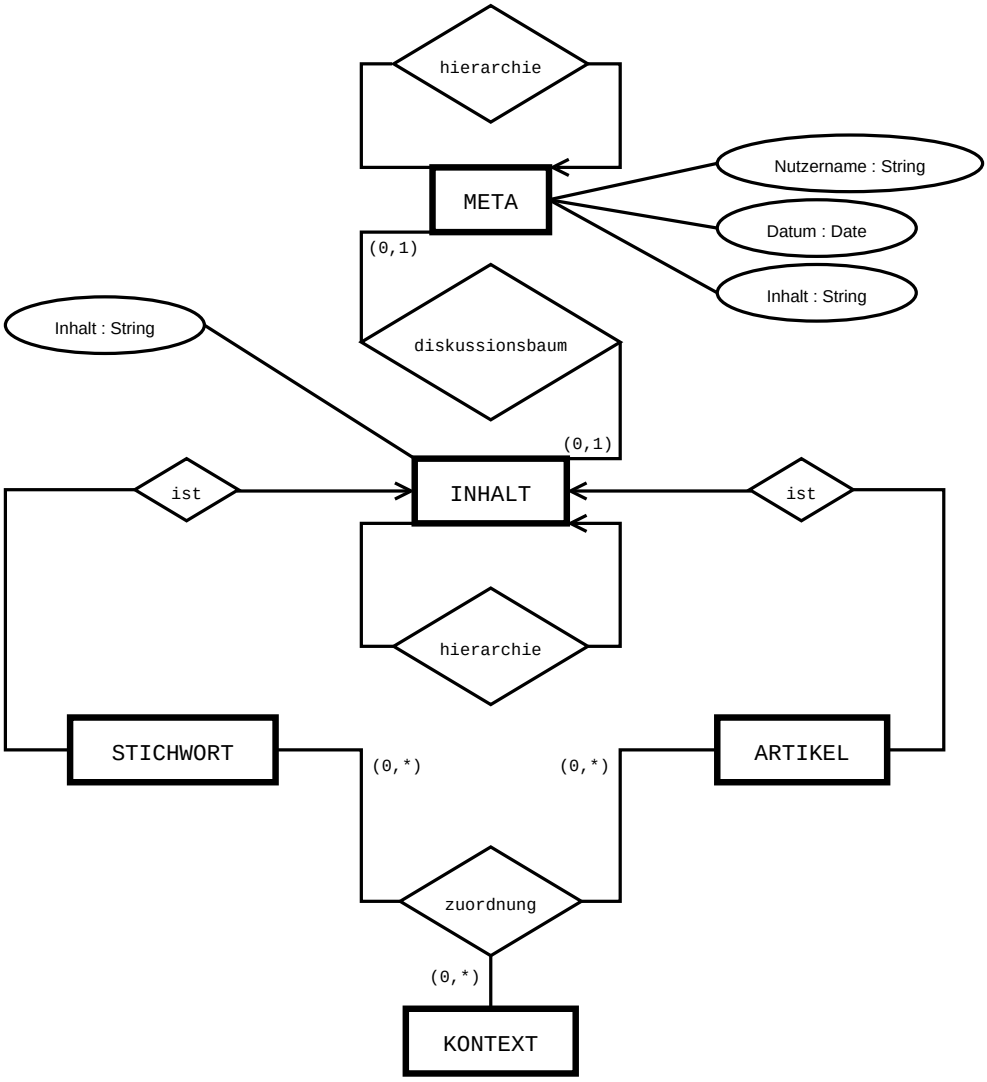

Abbildung 2.1: Entity-Relationship Diagramm

3 Integritätsbedingungen

In diesem Kapitel wird zunächst der Integritätsbegriff kurz geklärt, im Anschluß werden wesentliche Integritätsbedingungen der modellierten Datenbank genauer beleuchtet und abschließend einige Schlußbetrachtungen diesbezüglich angestellt.

3.1 Erläuterung

Während sich Datenbankdefinitionen mittels Entity-Relationship Diagramm (oder vergleichbaren Formalismen) im wesentlichen nur auf das DB-Schema beziehen, lassen sich mithilfe von Integritätsbedingungen Forderungen an den DB-Zustand formulieren, und somit vollständigere Spezifikationen der gewünschten Datenbank-Eigenschaften vornehmen. Integritätsbedingungen werden bei Einfüge-, Änderungs- und Löschoperationen (genauer: vor bzw. nach *Transaktionen*) durch das DBMS überprüft, und gewährleisten somit die Konsistenz einer Datenbank.

3.2 Formale Definition

Folgende Bedingungen stellen die Konsistenz der modellierten Datenbank sicher[1] (formuliert in einer, dem *Tupelkalkül* ähnlichen, Adhoc-Notation – da statt über die Schlüssel, welche im Entity-Relationship Diagramm noch nicht spezifiziert sind, über die Relationships argumentiert werden muß):

a) Beim Anlegen eines neuen `STICHWORT`-, oder `ARTIKEL`-Datum sind zwar aus technischer Sicht keine besonderen Bedingungen zu beachten, jedoch erscheint es sinnvoll zu fordern, daß zugleich obligatorisch ein `META`-Datum angelegt werden muß (in welchem sich der Autor widerspiegeln, und Quellenangaben, einordnende Kommentare, usw. befinden sollten):

$$\forall i : \texttt{INHALT} \left(\exists m : \texttt{META} \ (\texttt{diskussionsbaum}(i, m)) \right)$$

[1] Die Bedingungen beziehen sich auf das – in Abschnitt 2.3 definierte – formale Modell; auch korrektes Verhalten von Schlüssel- und Fremdschlüsselbeziehungen, sowie Einhaltung von Multiplizitätsangaben fällt unter das Thema Integrität, wird aber hier nicht weiter ausgeführt, weil es in der Regel standardmäßig sichergestellt ist.

b) Wird ein `META`-Datum angelegt, muß sichergestellt sein, daß es entweder einem anderen `META`-Datum (als „Kind“ im Diskussionsbaum), oder einem `INHALT`-Datum zugeordnet ist (aber nicht beidem). In Anbetracht von a) wäre es ausreichend, nur zu den ersten Teil der Bedingung zu fordern, jedoch deckt diese Formulierung auch späterer Integritätsbedingungen mit ab:

$$\begin{aligned}
&\forall m : \texttt{META}\ \Big(\\
&\quad (\exists m' : \texttt{META}\ (\texttt{hierarchie}\,(m, m')) \wedge \neg\,(\exists i : \texttt{INHALT}\ (\texttt{diskussionsbaum}\,(i, m)))) \\
&\qquad \vee \\
&\quad (\exists i : \texttt{INHALT}\ (\texttt{diskussionsbaum}\,(i, m)) \wedge \neg\,(\exists m' : \texttt{META}\ (\texttt{hierarchie}\,(m, m')))) \\
&\Big)
\end{aligned}$$

c) Außerdem ist eine sinnvolle Integritätsbedingung, daß das *Datum*-Attribut eines `META`-Datum nach dem seines Elter liegt:

$$\forall m, m' : \texttt{META}\ \Big(\texttt{hierarchie}\,(m, m') \rightarrow m.Datum < m'.Datum\Big)$$

d) Ein neues `KONTEXT`-Datum erfordert keine Integritätsbedingungen – es spricht nichts dagegen, ein `KONTEXT`-Datum anzulegen, ohne es einer `STICHWORT`/`ARTIKEL`-Beziehung zuzuordnen (wenngleich auch nichts dafür spricht).

e) Wird ein `STICHWORT`-Datum geändert, so stellt sich die Frage, ob nachher dieselben Zuordnungen (also verknüpfte `ARTIKEL`- bzw. `META`-Daten) wie vorher *semantisch* sinnvoll sind. Dies läßt sich jedoch nicht einfach in einen formale Integritätsbedingung übersetzen.

f) Für Änderungen an einem `ARTIKEL`-Datum gelten analoge Verhältnisse (bzgl. zugeordneter `STICHWORT`- bzw. `META`-Daten).

g) Wird ein `META`-Datum verändert, gilt natürlich weiterhin die Integritätsbedingung, daß das *Datum*-Attribut vor dem seiner Kinder bzw. nach dem seines Elter liegen muß (wodurch eine Änderung an einem `META`-Datum kaskadenartig Revisionen der zugeordneten Kinder erfordert): vgl. c); außerdem gilt Bedingung b) weiter.

Da sich aber Einträge in einem Diskussionsbaum explizit auf ihre Vorgänger (Eltern-Knoten) beziehen, und durch Änderungen an einem solchen Eintrag der Sinn der

nachfolgenden Beiträge entstellt werden könnte, scheint es sinnvoll, Änderungen an META-Daten generell zu verbieten[2]:

$$\begin{array}{l}\forall m : \texttt{META}_{alt} \ \big(\\ \quad \exists m' : \texttt{META}_{neu} \ \ (m' = m) \\ \big)\end{array}$$

h) Da ein KONTEXT-Datum nur zur Identifikation dient, kann sein Inhalt beliebig verändert werden.

i) Wenn ein STICHWORT- bzw. ARTIKEL-Datum gelöscht wird, muß auch der zugehörige META-Diskussionsbaum gelöscht werden: vgl. b).

j) Wenn ein STICHWORT- bzw. ARTIKEL-Datum gelöscht wird, müssen auch eventuelle Verweise aus der Taxonomie entfernt werden[3]:

$$\begin{array}{l}\forall i : \texttt{INHALT}_{alt} \ \big(\\ \quad \neg \left(\exists i' : \texttt{INHALT}_{neu} \ \ (i' = i)\right) \rightarrow \\ \qquad \neg \left(\exists i' : \texttt{INHALT}_{neu} \ \ (\texttt{hierarchie}_{neu}\,(i', i) \vee \texttt{hierarchie}_{neu}\,(i, i'))\right) \\ \big)\end{array}$$

k) Wenn ein STICHWORT- bzw. ARTIKEL-Datum gelöscht wird, müssen auch die Zuordnungen zu ARTIKEL- bzw. STICHWORT-Daten gelöscht werden – analog zu j):

$$\begin{array}{l}\forall s : \texttt{STICHWORT}_{alt} \ \big(\\ \quad \neg \left(\exists s' : \texttt{STICHWORT}_{neu} \ \ (s' = s)\right) \rightarrow \neg \left(\exists a : \texttt{ARTIKEL}_{neu} \ \ (\texttt{zuordnung}_{neu}\,(s, a))\right) \\ \big)\end{array}$$

$$\begin{array}{l}\forall a : \texttt{ARTIKEL}_{alt} \ \big(\\ \quad \neg \left(\exists a' : \texttt{ARTIKEL}_{neu} \ \ (a' = a)\right) \rightarrow \neg \left(\exists s : \texttt{STICHWORT}_{neu} \ \ (\texttt{zuordnung}_{neu}\,(s, a))\right) \\ \big)\end{array}$$

l) Wird ein META-Datum entfernt, müssen auch dessen Kinder entfernt werden: vgl. b).

[2] Zu diesem Zweck muß ein neues Konzept eingeführt werden, um DB-Zustände vor bzw. nach Transaktionen miteinander vergleichen zu können. Ausdrücke die sich auf einen DB-Zustand *vor* der Transaktion beziehen sollen, werden mit einem Subscript $_{alt}$ versehen; soll sich ein Ausdruck auf den DB-Zustand *nach* der Transaktion beziehen, versieht man ihn mit dem Subscript $_{neu}$.

[3] Diese Integritätsbedingung ist eigentlich ein Vorgriff auf die technische Umsetzung von Relationships. Es wird trotzdem ein (syntaktisch nicht ganz einwandfreier) Ausdruck angegeben, der in etwa die gewünschte Eigenschaft widerspiegelt.

m) Ein KONTEXT-Datum sollte nur dann gelöscht werden, wenn es nicht (mehr) in STICHWORT/ARTIKEL-Zuordnungen referenziert wird:

$$\forall k : \texttt{KONTEXT}_{alt} \; \Big(\\ \neg \left(\exists k' : \texttt{KONTEXT}_{neu} \;\; (k' = k)\right) \rightarrow \neg \left(\texttt{zuordnung}_{neu}\,(k)\right) \\ \Big)$$

3.3 Reflexion

Die angegebenen Integritätsbedingungen wurden systematisch ermittelt (Einfügen, Ändern, Löschen der modellierten Entitäten betrachtet), dennoch ist damit nicht sichergestellt, daß prinzipiell alle sinnvollen Integritätsbedingungen angegeben sind.

Selbstverständlich dürfen Integritätsbedingung einander nicht widersprechen, da ansonsten kein fehlerfreier DB-Zustand erreicht werden kann. Aus redundante Bedingungen (d. h. Ausdrücke, die sich inhaltlich überschneiden) kann der gemeinsame Anteil extrahiert und als separate Integritätsbedingung angegeben werden.

Grundsätzlich ist es schwierig, für unstrukturierte Daten Integritätsbedingungen anzugeben, wie sich z. B. an den Bedingungen e) und f) zeigt.

Die Integrationsbedingung g) verweigert in ihrer derzeitigen Formulierung auch das Löschen von META-Daten, ist also eigentlich zu stark. Allerdings sind an dieser Stelle noch keine differenzierteren Sprachmittel zur adäquaten Beschreibung verfügbar. Diese Schwäche wird in Abschnitt 4.1.5 behoben.

4 Synthese des relationalen Schema

Dieses Kapitel befaßt sich damit, das formal spezifizierte Datenmodell in entsprechende relationale Schemata zu überführen. Dazu wird zunächst das Entity-Relationship-Modell (vgl. Abschnitt 2.3) in ein Relationen-Modell übersetzt, welches anschließend nach dem sog. Standardverfahren normalisiert wird; schließlich sind noch einige Besonderheiten der, in Abschnitt 3.2 eingeführten, Integritätsbedingungen zu betrachten.

4.1 Standardverfahren

Zuerst werden Primärschlüssel festgelegt, anschließend sind die Entitätstypen und danach die Beziehungstypen des Entity-Relationship-Modell in Relationenschemata zu übersetzen, wobei hierfür die zuvor definierten Schlüssel eingesetzt werden.

`META`: Da die Attribute *Nutzername* und *Datum* (= Datums- und Zeitangabe) den *Inhalt* determinieren sollten, werden als Schlüsselattribute *Nutzername* und *Datum* ausgewählt.

`INHALT`: Dieser Entitätstyp fungiert als Basis für die abgeleiteten Entitätstypen `STICHWORT` und `ARTIKEL`, er wird demzufolge nur in seinen Spezialfällen betrachtet.

`STICHWORT`: Das einzige (von `INHALT` geerbte) Attribut *Inhalt* bestimmt die Entitäten dieses Typs, ist also Schlüsselattribut.

`ARTIKEL`: Analog zu `STICHWORT` bestimmt *Inhalt* die Entitäten dieses Typs, ist also Schlüsselattribut.

`KONTEXT`: Diesem Entitätstyp sind bislang gar keine Attribute zugewiesen, es muß also ein Schlüsselattribut (mit Namen *Id*) neu angelegt werden.

Es folgen die Schemata für die Entitätstypen des Entity-Relationship-Modell:

`META` (*Nutzername* , *Datum* , *Inhalt*)

`STICHWORT` (*Inhalt*)

`ARTIKEL` (*Inhalt*)

`KONTEXT` (*Id*)

Nun folgen die Schemata für Beziehungstypen des Entity-Relationship-Modell, wobei für funktionale Beziehungen kein eigenes relationales Schema angelegt wird, sondern das Schema einer beteiligten Entität verändert wird:

`hierarchie: META` (*Nutzername* , *Datum* , *Inhalt* , *ElterNutzername* , *ElterDatum*)

`diskussionsbaum: STICHWORT` (*Inhalt* , *MetaNutzername* , *MetaDatum*)

`diskussionsbaum: ARTIKEL` (*Inhalt* , *MetaNutzername* , *MetaDatum*)

`hierarchie: STICHWORT` (*Inhalt* , *MetaNutzername* , *MetaDatum* , *ElterInhalt*)

`hierarchie: ARTIKEL` (*Inhalt* , *MetaNutzername* , *MetaDatum* , *ElterInhalt*)

`zuordnung` (*StichwortInhalt* , *ArtikelInhalt* , *KontextIds* ,)

Das initiale Relationenschema ist also:

- `META` (*Nutzername* , *Datum* , *Inhalt* , *ElterNutzername* , *ElterDatum*)
- `STICHWORT` (*Inhalt* , *MetaNutzername* , *MetaDatum* , *ElterInhalt*)
- `ARTIKEL` (*Inhalt* , *MetaNutzername* , *MetaDatum* , *ElterInhalt*)
- `KONTEXT` (*Id*)
- `zuordnung` (*StichwortInhalt* , *ArtikelInhalt* , *KontextIds* ,)

In dieser Form treten aber noch massenhaft *Änderungsanomalien* auf (d. h. wenn ein Datenbankeintrag geändert wird, müssen > 1 Einträge geändert werden), eine weitere Anpassung des Relationenschema ist daher wünschenswert. Die Einzelschritte zur Transformation des Relationenschema liegen auf der Hand und werden an dieser Stelle weggelassen (im wesentlichen Einführen neuer Schlüsselattribute und darauf referenzieren, ggf. auch erzeugen eines neuen Relationenschema):

- `NUTZER` (*Id* , *Name*)
- `META` (*Id* , *NutzerId* , *Datum* , *Inhalt* , *ElterId*)
- `STICHWORT` (*Id* , *Inhalt* , *MetaId* , *ElterId*)
- `ARTIKEL` (*Id* , *Inhalt* , *MetaId* , *ElterId*)
- `KONTEXT` (*Id*)
- `zuordnung` (*StichwortId* , *ArtikelId* , *KontextIds* ,)

Aus semantischen Gründen drängt sich eine weitere Vereinfachung auf. Da die `KONTEXT`-Entitäten lediglich zur Identifikation gemeinsamer Kontexte von Zuordnungen dienen, sich bloße Id-Zahlen aber mnemotechnisch schlecht handhaben lassen, bieten sich die `STICHWORT`-Entitäten geradezu als Kontext-Referenzen an. Eine neuerliche Transformation führt also zu folgendem Relationenschema:

- `NUTZER` (*Id* , *Name*)
- `META` (*Id* , *NutzerId* , *Datum* , *Inhalt* , *ElterId*)
- `STICHWORT` (*Id* , *Inhalt* , *MetaId* , *ElterId*)
- `ARTIKEL` (*Id* , *Inhalt* , *MetaId* , *ElterId*)
- `zuordnung` (*StichwortId* , *ArtikelId* , *KontextStichwortIds* ,)

4.1.1 erste Normalform

Damit alle Attributwerte atomar sind, muß die `zuordnung`-Relation verändert werden (da im Attribut *KontextStichwortIds* bisher eine Wertemenge gespeichert wird).

- `zuordnung` (*StichwortId* , *ArtikelId* , *KontextStichwortId* ,)

4.1.2 zweite Normalform

Es hängt kein Nichtschlüsselattribut nur partiell von einem Schlüssel ab (d. h. jedes Nichtschlüsselattribut hängt jeweils vom gesamten Schlüssel ab), also genügt das Relationenschema der zweiten Normalform.

4.1.3 dritte Normalform

Da kein Nichtschlüsselattribut transitiv von einem Schlüssel abhängt (d. h. es existieren keine funktionalen Abhängigkeiten zwischen Nichtschlüsselattributen), genügt das Schema auch der dritten Normalform.

4.1.4 weitere Normalisierungen

Es existieren noch weitere Kriterien einer „guten“ Datenbank-Struktur[1] – aufgrund der überschaubaren Größe der hier modellierten Datenbank, kann aber auf eine Fortführung des Verfahrens verzichtet werden.

Die modellierte Datenbank wird also schlußendlich mithilfe der folgenden Relationenschemata realisiert:

[1] wie z. B. die *BCNF* (Boyce-Codd-Normalform), oder *4NF* (vierte Normalform)

- NUTZER (*Id* , *Name*)
- META (*Id* , *NutzerId* , *Datum* , *Inhalt* , *ElterId*)
- STICHWORT (*Id* , *Inhalt* , *MetaId* , *ElterId*)
- ARTIKEL (*Id* , *Inhalt* , *MetaId* , *ElterId*)
- zuordnung (*StichwortId* , *ArtikelId* , *KontextStichwortId* ,)

4.1.5 Integritätsbedingungen

Nachdem nun die relationale Struktur der Datenbank feststeht, können auch die Integritätsbedingungen (vgl. 3.2) entsprechend konkretisiert werden.

Die, in der formalen Beschreibung der Integritätsbedingungen verwendeten, Prädikate lassen sich folgendermaßen übersetzen:

a), b)

$$\begin{aligned}&\forall i : \texttt{STICHWORT} \cup \texttt{ARTIKEL},\ m : \texttt{META}\ (\\ &\quad \texttt{diskussionsbaum}\,(i, m) \quad := \quad (i.MetaId = m.Id)\\ &)\end{aligned}$$

b), c)

$$\begin{aligned}&\forall m, m' : \texttt{META}\ (\\ &\quad \texttt{hierarchie}\,(m, m') \quad := \quad (m.ElterId = m'.Id)\\ &)\end{aligned}$$

j)

$$\begin{aligned}&\forall i, i' : \texttt{STICHWORT} \cup \texttt{ARTIKEL}\ (\\ &\quad \texttt{hierarchie}\,(i, i') \quad := \quad (i.ElterId = i'.Id)\\ &)\end{aligned}$$

k)

$$\begin{aligned}&\forall s : \texttt{STICHWORT},\ a : \texttt{ARTIKEL}\ \exists z : \texttt{zuordnung}\ (\\ &\quad \texttt{zuordnung}\,(s, a) \quad := \quad (z.StichwortId = s.Id \wedge z.ArtikelId = a.Id)\\ &)\end{aligned}$$

m)

$$\begin{aligned}&\forall k : \texttt{STICHWORT}\ \exists z : \texttt{zuordnung}\ (\\ &\quad \texttt{zuordnung}\,(k) \quad := \quad (z.KontextStichwortId = k.Id)\\ &)\end{aligned}$$

Auch die Bedingung g) läßt sich nun in einer adäquaten Form angeben (vgl. Abschnitt 3.3):

$$\begin{array}{l} \forall m : \texttt{META}_{alt} \ (\\ \quad \exists m' : \texttt{META}_{neu} \ \ (m'.Id = m.Id \rightarrow m' = m) \\ ) \end{array}$$

Um Mehrdeutigkeiten und Interpretationsproblemen bei der Auswertung von Anfragen (vgl. Kapitel 6) vorzubeugen, sind noch folgende Eindeutigkeitsbedingungen hilfreich:

$$\forall n, n' : \texttt{NUTZER} \ \big(n'.Name = n.Name \rightarrow n' = n\big)$$

$$\forall s, s' : \texttt{STICHWORT} \ \big(s'.Inhalt = s.Inhalt \rightarrow s' = s\big)$$

$$\forall a, a' : \texttt{ARTIKEL} \ \big(a'.Inhalt = a.Inhalt \rightarrow a' = a\big)$$

Mithilfe dieser Angaben könnte die entworfenen Datenbank nun in einem relationalen DBS angelegt (und überwacht) werden. Schwerpunkt dieser Ausarbeitung ist allerdings nicht der Betrieb, sondern der Entwurf einer Datenbank – daher endet das Kapitel an dieser Stelle.

5 Sichten

Bekanntermaßen lassen sich für unterschiedliche Verwendungen einer DB verschiedene externe Sichten definieren. In diesem Kapitel werden einige mögliche Sichten exemplarisch angelegt, dabei kommt als Beschreibungssprache SQL[1] zum Einsatz.

Datenbasis:

```
create view ZUORDNUNG (Stichwort, Artikel, Kontext)
as select S.Inhalt, A.Inhalt, K.Inhalt
   from STICHWORT S, ARTIKEL A, STICHWORT K, zuordnung z
   where S.Id=z.StichwortId
     and A.Id=z.ArtikelId
     and K.Id=z.KontextStichwortId
```

Diese Sicht erlaubt einen intuitiven Zugriff auf die Inhaltsdaten der DB, wobei jeweils zugeordnete Stichwörter/Artikel zusammen mit ihren, je Zuordnung definierten, Kontexten angezeigt werden.

Übersicht der Benutzerbeiträge (ohne Metadaten):

```
create view BENUTZERINHALT (Inhalt, Nutzername, Datum)
as (select S.Inhalt, Name, Datum
    from STICHWORT S, META, NUTZER
    where META.Id=S.MetaId
      and NUTZER.Id=META.NutzerId)
   union
   (select A.Inhalt, Name, Datum
    from ARTIKEL A, META, NUTZER
    where META.Id=A.MetaId
      and NUTZER.Id=META.NutzerId)
```

Mittels diese Sicht erhält man einen direkten Überblick, wann welcher inhaltliche Betrag von welchem Nutzer verfaßt wurde.

[1] **S**tructured **Q**uery **L**anguage

Statistik aller Beiträge je Nutzer:

```
create view STATISTIK-STICHWORT (Nutzername, Anzahl)
as select Name, count(distinct S.Id)
   from STICHWORT S, META, NUTZER
   where META.Id=S.MetaId
     and NUTZER.Id=META.NutzerId

create view STATISTIK-ARTIKEL (Nutzername, Anzahl)
as select Name, count(distinct A.Id)
   from ARTIKEL A, META, NUTZER
   where META.Id=A.MetaId
     and NUTZER.Id=META.NutzerId

create view STATISTIK-INHALT (Nutzername, Anzahl)
as select Nutzername, sum(Anzahl)
   from ((select * from STATISTIK-STICHWORT)
         union
         (select * from STATISTIK-ARTIKEL))
   group by Nutzername

create view STATISTIK-META (Nutzername, Anzahl)
as select Name, count(distinct M.Id)
   from META M, NUTZER
   where NUTZER.Id=META.NutzerId

create view STATISTIK-GESAMT (Nutzername, Anzahl)
as select Nutzername, sum(Anzahl)
   from ((select * from STATISTIK-INHALT)
         union
         (select * from STATISTIK-META))
   group by Nutzername
```

Die o. a. fünf Sichten ermöglichen einen direkten Zugriff auf die Statistiken der im System erfaßten Beiträge, aufgeschlüsselt nach Benutzernamen.

Taxonomien:

```
create view TAXONOMIE-STICHWORT (Stichwort, Kind)
as select Elter.Id, Kind.Id
   from STICHWORT Elter, STICHWORT Kind
   where Elter.Id=Kind.ElterId
```

```
create view TAXONOMIE-ARTIKEL (Artikel, Kind)
as select Elter.Id, Kind.Id
   from ARTIKEL Elter, ARTIKEL Kind
   where Elter.Id=Kind.ElterId
```

Die oben aufgeführten Sichten listen jeweils die taxonomische Hierarchie der Stichwort- bzw. Artikel-Relation, in Form von Schlüssel-Zuordnungen, auf.

Es lassen sich noch diverse weitere (sinnvollere und weniger sinnvolle) externe Sichten auf die modellierte DB angeben, aber um das Konzept zu illustrieren dürften die aufgeführten Beispiele ausreichen.

6 Standardanfragen

Um den Entwurfsprozeß abzurunden, werden in diesem Kapitel – beispielhaft – einige typische DB-Anfragen formuliert. Die Anfragen werden mithilfe unterschiedlicher Formalismen ausgedrückt, um eine Vergleichsmöglichkeit zu bieten.

Folgende Beispielanfragen sind zu übersetzen:

a) „Liefere mir alle Artikel (a) zu einem bestimmten Stichwort (s) zurück, die diesem im Kontext (k) zugeordnet sind!“

b) „Liefere mir die Metainformationen (m) zu einem bestimmten Artikel (a)!“

c) „Gib mir alle Kinder (m') eines bestimmten Metadatum (m) zurück!“

d) „Zeige mir ein Liste aller Kontexte (k), die einem bestimmten Stichwort/Artikel-Paar (s/a) zugeordnet sind!“

6.1 SQL

a)
```
select A.Id, A.Inhalt
from ARTIKEL A, zuordnung z, STICHWORT S, STICHWORT K
where S.Inhalt='s' and K.Inhalt='k'
  and z.StichwortId=S.Id and z.KontextStichwortId=K.Id
  and A.Id=z.ArtikelId
```

b)
```
select Name, Datum, M.Inhalt
from NUTZER N, META M, ARTIKEL A
where A.Id='a'
  and M.Id=A.MetaId and N.Id=M.NutzerId
```

c)
```
select Name, Datum, Inhalt
from META, NUTZER
where ElterId='m' and NUTZER.Id=NutzerId
```

d)
```
select K.Inhalt
from STICHWORT S, ARTIKEL A, zuordnung z, STICHWORT K
where S.Inhalt='s' and A.Inhalt='a'
  and z.StichwortId=S.Id and z.ArtikelId=A.Id
  and K.Id=z.KontextStichwortId
```

6.2 RA[1]

a)

$$\pi_{\texttt{ARTIKEL}.Id,\texttt{ARTIKEL}.Inhalt}(\texttt{ARTIKEL} *_{Id=ArtikelId} ((\sigma_{Inhalt=\text{'s'}}(\texttt{STICHWORT}) *_{Id=StichwortId} \texttt{zuordnung}) *_{KontextStichwortId=Id} \sigma_{Inhalt=\text{'k'}}(\texttt{STICHWORT})))$$

b)

$$\pi_{Name,Datum,\texttt{META}.Inhalt}((\texttt{NUTZER} *_{Id=NutzerId} \texttt{META}) *_{\texttt{META}.Id=MetaId} \sigma_{Id=\text{'a'}}(\texttt{ARTIKEL}))$$

c)

$$\pi_{Name,Datum,Inhalt}(\texttt{NUTZER} *_{Id=NutzerId} \sigma_{ElterId=\text{'m'}}(\texttt{META}))$$

d)

$$\pi_{\texttt{STICHWORT}.Inhalt_1}(\texttt{STICHWORT} *_{Id=KontextStichwortId} ((\sigma_{Inhalt=\text{'s'}}(\texttt{STICHWORT}) *_{Id=StichwortId} \texttt{zuordnung}) *_{ArtikelId=Id} \sigma_{Inhalt=\text{'a'}}(\texttt{ARTIKEL})))$$

6.3 BK[2]

a)

$$\{aId, aInhalt \mid \exists sId, sInhalt, kId, kInhalt\ (\texttt{STICHWORT}(sId, sInhalt, *, *) \wedge \texttt{STICHWORT}(kId, kInhalt, *, *) \wedge \texttt{ARTIKEL}(aId, aInhalt, *, *) \wedge \texttt{zuordnung}(sId, aId, kId) \wedge sInhalt = \text{'s'} \wedge kInhalt = \text{'k'})\}$$

b)

$$\{nName, mDatum, mInhalt \mid \exists aId, mId, nId\ (\texttt{ARTIKEL}(aId, *, mId, *) \wedge \texttt{META}(mId, nId, mDatum, mInhalt, *) \wedge \texttt{NUTZER}(nId, nName) \wedge aId = \text{'a'})\}$$

c)

$$\{nName, mDatum, mInhalt \mid \exists mElter, nId\ \texttt{META}(*, nId, mDatum, mInhalt, mElter) \wedge \texttt{NUTZER}(nId, nName) \wedge mElter = \text{'m'})\}$$

d)

$$\{kInhalt \mid \exists sId, sInhalt, aId, aInhalt, kId\ (\texttt{STICHWORT}(sId, sInhalt, *, *) \wedge \texttt{ARTIKEL}(aId, aInhalt, *, *) \wedge \texttt{zuordnung}(sId, aId, kId) \wedge \texttt{STICHWORT}(kId, kInhalt, *, *) \wedge sInhalt = \text{'s'} \wedge aInhalt = \text{'a'})\}$$

[1] **R**elationen**a**lgebra
[2] **B**ereichs**k**alkül

7 Schluß

Diese Ausarbeitung ist bewußt in einem gedrängten Stil verfaßt worden, um trotz der Stoffdichte eine gewisse Seitenzahl nicht zu überschreiten. Es wurden daher keine lehrbuchartigen Einführungen in die verwendeten Begriffe gegeben, stattdessen sind die im Rahmen der Vorlesung eingeführten Konzepte direkt angewandt worden. Die Ausarbeitung richtet sich folglich an einen in DBS schon erfahrenen Leser.

erweiterte Modellierung

Um den Entwurfsprozeß nicht unnötig zu verkomplizieren, sind einige Vereinfachungen am Datenmodell vorgenommen worden[1]. Ein realitätsnäheres Modell ist in Abb. 7.1 angegeben.

Ergebnissortierung

Da die Möglichkeit, zu einem Stichwort beliebig viele Artikel (und umgekehrt) angeben zu können, über kurz oder lang ein Überangebot an Treffern je Suchanfrage bewirkt, erscheint es sinnvoll die Suchergebnisse nach Relevanz zu ordnen.

Dazu werden die (je Nutzer individuellen) Bewertungen der Inhalte und Zuordnungen interpretiert, wobei ausgenutzt wird, daß jeder Benutzer auch andere Benutzer beurteilen kann (und so deren Bewertungen indirekt mitverwendet).

Ein Algorithmus für (bewertungsprofilabhängige) Ergebnissortierung müsste also ausgehend von einem Nutzer ABC zu jedem Suchergebnis XYZ ein „Vorurteil“ bzw. eine Relevanz-Maßzahl ermitteln (vgl. Abb. 7.2).

[1] Es wurde z. B. die Möglichkeit weggelassen, das Gelten einer Stichwort/Artikel-Zuordnung je Benutzer und Kontext individuell festzulegen (vgl. Abschnitt 2.1). Desweiteren wäre auch eine Bewertungsfunktion der Inhalte selbst (nicht nur deren Zuordnungen) denkbar. Schließlich könnte die Taxonomie statt als funktionale auch als n-zu-m Beziehung modelliert werden, u. a. m.

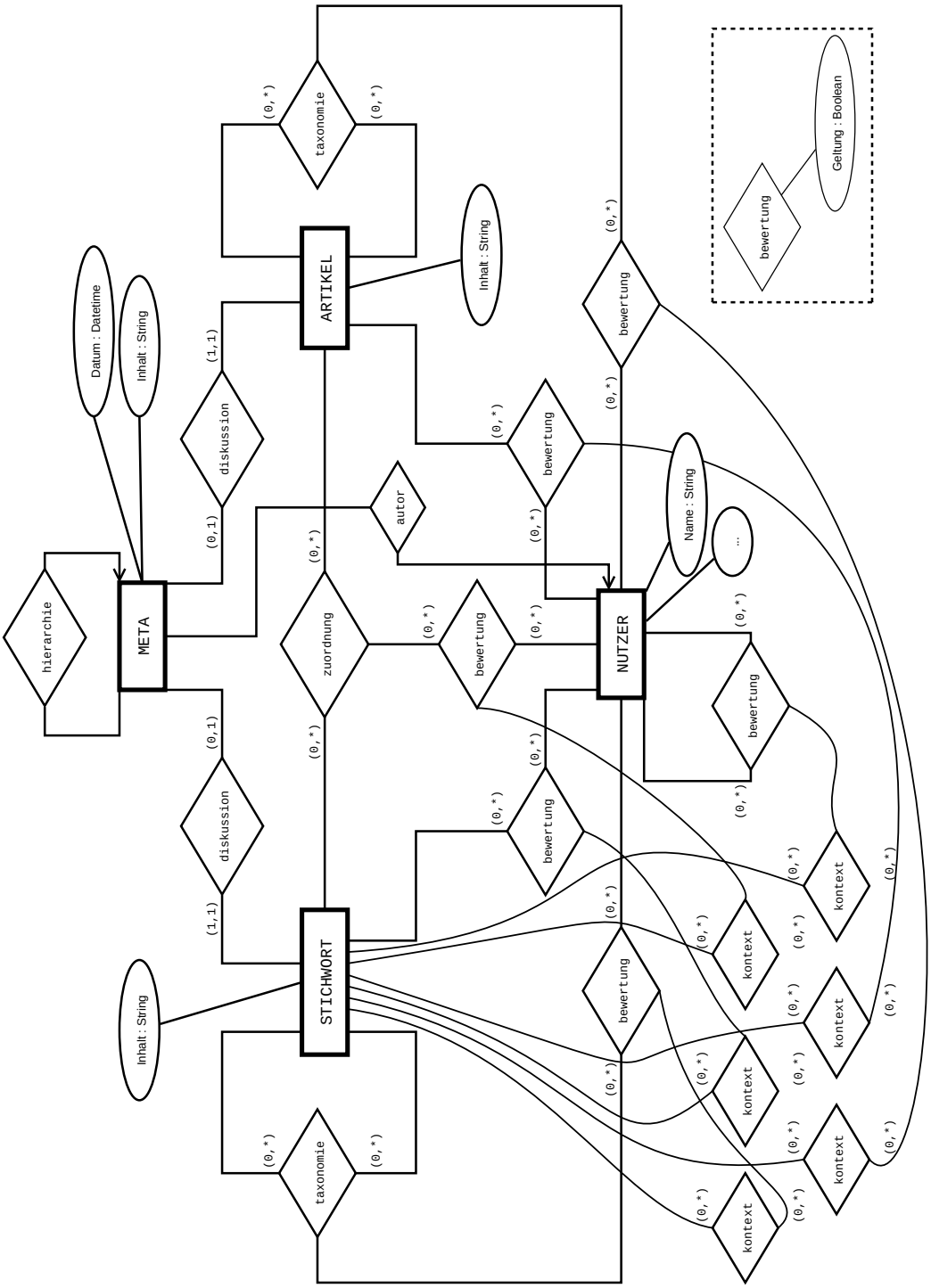

Abbildung 7.1: Entity-Relationship Diagramm – realitätsnäher

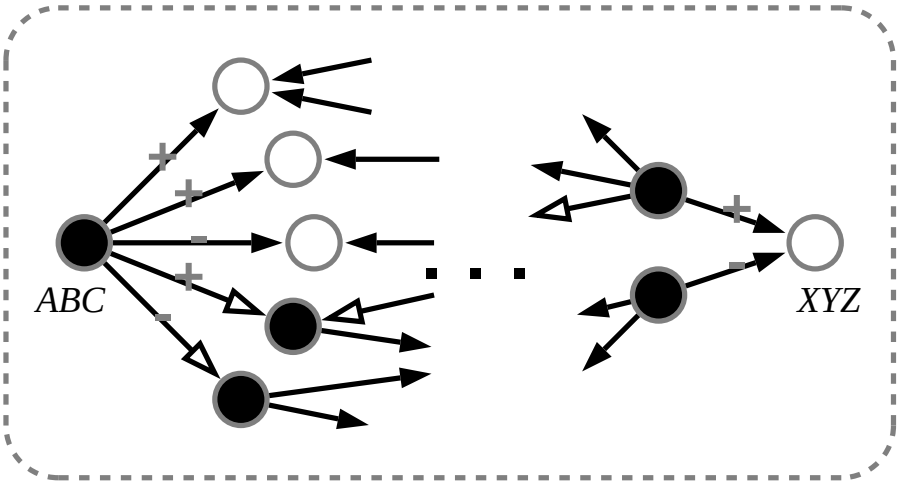

Abbildung 7.2: Bewertungsfunktion von Nutzer *ABC* nach Datum *XYZ*

Literaturverzeichnis

[1] Prof. Dr. M. Gogolla; „Material zur Vorlesung“, Lehrveranstaltung: Datenbanksysteme, Wintersemester 2008/9, Uni-Bremen, http://db.informatik.uni-bremen.de/teaching/courses/ws2008_dbs/

[2] A. Darabos, C. Duddeck, R. Eckert, S. Klagge, J.-C. Kranefeld, R. Rothaupt, P. Taffou; „Beispielausarbeitungen“, (im Rahmen der LV-Datenbanksysteme zur Verfügung gestellt)

[3] WIKIPEDIA; „diverse Artikel“, http://de.wikipedia.org/wiki/ANSI-SPARC-Architektur, http://de.wikipedia.org/wiki/Konsistenz_(Datenbank), http://de.wikipedia.org/wiki/Integrit%C3%A4tsbedingung, http://de.wikipedia.org/wiki/Relationale_Algebra, http://de.wikipedia.org/wiki/Kalk%C3%BCl_(Datenbank)

[4] H. Kreissl; „Einführung Datenbanktechnik und Entwurf“ (2002-2005), http://www.kreissl.info/diggs/db_inhalt.php

[5] O. Müller, O. Vornberger; „Datenbanksysteme“, Vorlesung aus dem Sommersemester 2001, Uni-Osnabrück http://www-lehre.informatik.uni-osnabrueck.de/~dbs/2001/skript/

[6] U. Kelter; „diverse Lehrmaterialien zum Thema: Datenbanksysteme“, (z. B. dmer, ers, rdbm, rtk, tae, tid, ...) http://pi.informatik.uni-siegen.de/kelter/lehre/lm/

*Alle Online-Zugriffe im Februar 2009 *